

A Proposed Methodology for Performing Compliance Validation Configuration Auditing

**Prepared by:
Dr. Bruce Gabrielson**

**SAIC Center for Information Security
Technology
7125 Columbia Gateway Dr, Suite 300
Columbia, Maryland 21046**

**In association with:
Ms. Penny Klein**

**Defense Information Systems Agency, D25
5111 Leesburg Pike, Suite 400
Falls Church, VA 22041-3206**

Summary

This paper addresses the results of a study examining potential techniques or methods that could be used to perform configuration audits in support of the DITSCAP [1] Phase 4, Post Accreditation process. Using system changes (considered trigger events) that might result in a potential vulnerability, the study focused on how to obtain information about the change rather than the vulnerability itself. The purpose of compliance validation (CV) activities are to ensure the continued compliance with the security requirements, current threat assessment, and concept of operations as stated and agreed upon in the SSAA [2]. The proposed approach presents a “snapshot-based” method to perform CV configuration auditing. Specifically, a technique is proposed that will quickly provide real time change information sufficient to indicate the need for further security related testing.

Introduction

The introduction of the DITSCAP has proven to be a useful approach to satisfy accreditation requirements for Government Information Technology Systems (ITSs). It is tailored to provide a complete solution in a climate with constantly changing multiple complex systems. DITSCAP Phase 4, Post Accreditation, includes those activities (shown in Figure 1) necessary for the continuing operation of the accredited IS in its specified computing environment. This phase starts after the system has been

certified and accredited for operations. The objectives of Phase 4 are to ensure secure system management, operation, and maintenance to preserve an acceptable level of residual risk. The methodology for performing this function is called Compliance Validation.

The objective of the Compliance Validation and Inspection (CVI) process is (1) to assess the degree to which Defense Information System Agency’s (DISA) information systems (ISs) comply with Department of Defense (DoD), Office Of Management and Budget (OMB), and National Security Requirements and (2) to recommend a level of validation testing that will maintain information system accreditation in accordance with DoD security guidance. The challenge for DoD is to implement a CVI program that can keep the security posture current, be responsive to changing missions, environments, and architectures, be supportable by scarce resources, and can maintain the information system’s accreditation.

After an IS is approved for operation in a specific computing environment, changes to the information system and the computing environment must be continuously monitored and controlled. Although changes can adversely affect the overall security posture of the infrastructure and the information system, change is also a necessary response to evolving misuse, users’ needs, and new technology developments.

Accreditation using the DITSCAP process is currently based upon a formal security validation. This is performed through testing and an evaluation that ties each system's certified hardware and software to the configuration of the computing environment as well as the relationship of other technologies versus the common infrastructure at the time of the accreditation. Subsequent changes in the information system configuration, operational mission, computing environment or the computing environment's configuration may invalidate the security posture identified during the initial accreditation. The ability and timeliness of the accrediting organization to determine when a change has been made is critical to continued success of the overall secure posture.

revalidation within the DITSCAP process are initially identified to the accrediting organization based on a defined condition. This condition can be the result of a request notification, the system has exceeded a periodic schedule for revalidation, or the system has exhibited characteristics that could indicate a security problem. Periodically scheduled systems are default candidates for accreditation, while behavior related re-validations are based on what is defined as triggering events.

Any characteristics that could indicate a security problem might exist are called triggering events. A known modification to any portion of the system mission, environment, or architecture that affects the system's overall security posture would be a triggering event. CV triggering events signal a potential change in the accreditation

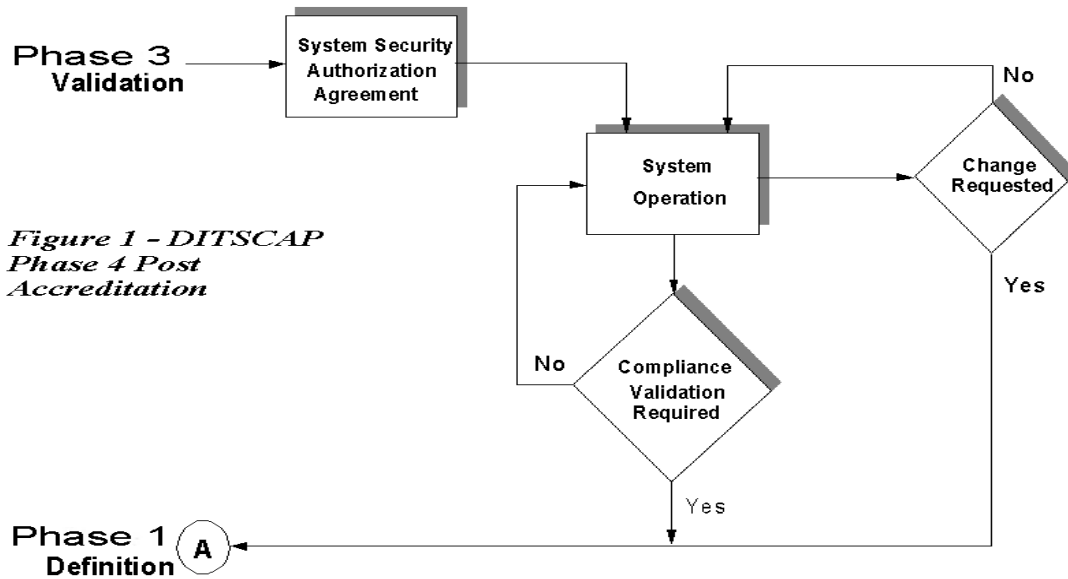


Figure 1 - DITSCAP Phase 4 Post Accreditation

Background on Events That Indicate Revalidation

The DITSCAP recognizes the evolving nature of information systems and technologies and through change management attempts to address this evolution during Phase 4. Systems requiring

posture of the system has taken place. The trigger events identified include:

- Change requests/notifications – new system or connected systems
- Configuration Audit Indicating a Changed Configuration
- Personnel Changes

<i>Identified Change</i>	<i>Major</i>	<i>Minor</i>
<i>1. Operating System</i>	<i>X</i>	
<i>2. Primary Application</i>	<i>X</i>	
<i>3. Adding Another Major Application</i>	<i>X</i>	
<i>4. Policy/Mission Needs Change</i>	<i>X</i>	
<i>5. Adding Another Service</i>	<i>X</i>	
<i>6. Environment Change</i>		<i>X</i>
<i>7. Logical/Physical Change</i>		<i>X</i>
<i>8. Equipment Upgrade</i>		<i>X</i>
<i>9. Security Application Change</i>		<i>X</i>

Table 1 – Configuration Changes

- Threat Documentation
- New Security Advisory
- Trouble/Vulnerability Test Reports

Note that while some of the above trigger events obviously indicate a potential security problem requiring immediate security organization action, some simply indicate that a change has occurred. These changes are changes that are seldom acted upon or evaluated immediately and do not always result in a scheduled CV test.

Changes That Might Indicate a Vulnerability

Testing experience provides a good feel for the types and severity of changes that may cause a vulnerability problem. These changes are shown in Table 1 [3]. Those listed as “minor” are indicated as such only because they are less often the source of a vulnerability problem, not because the vulnerability is less severe in nature. Of concern is that not every identified type of change can be readily detected during testing using available remote vulnerability test tools or other documentation that might be provided.

Detecting Change and Resource Allocation

When considering the ability to detect a change, many indicators are taken into consideration. However, two sources of detection concern are noted. First, there might be some vulnerability that would be easy to detect using host based only approaches. Second, it is possible that a significant vulnerability related change might remain undetected until a revalidation simply because it isn’t looked for remotely during normal vulnerability testing. Current remote vulnerability test tools look for problems, not internal changes.

Organizations that support the accreditation process also have the problem of applying their critical test resources in a timely manner where they are most needed. It is expensive to continually send test personnel to user’s sites to test for vulnerabilities that may or may not exist. The DITSCAP currently details a robust change management process on a three-year re-accreditation cycle. Continuing technology improvements can cause rapid changes in the installed computing base as depicted in Figure 2. The top broken line indicates changes taking place at shorter than three-year intervals. The middle CV line indicates a regular cycle of configuration audit testing that would closely monitor and detect when

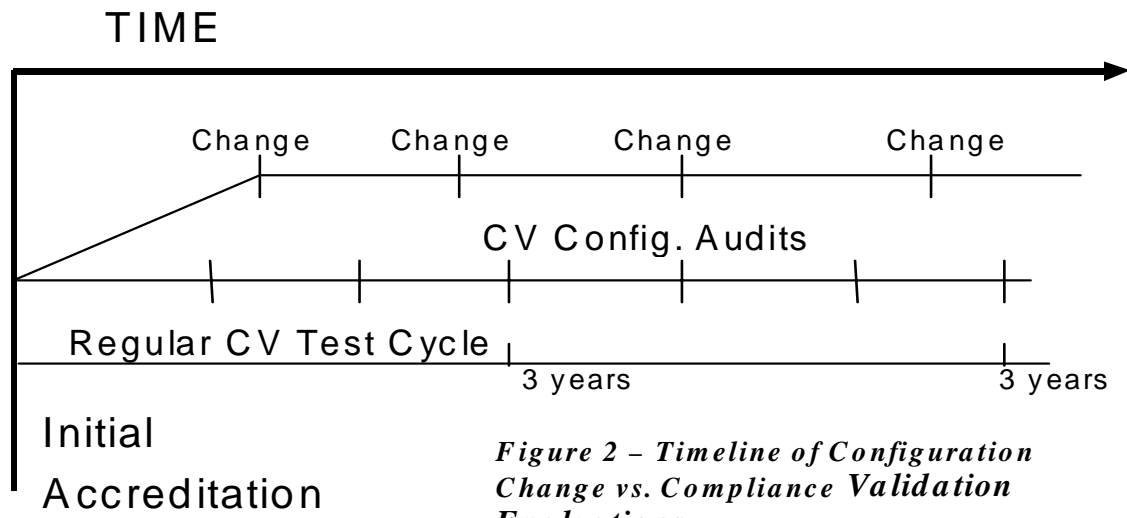


Figure 2 – Timeline of Configuration Change vs. Compliance Validation Evaluations

changes take place. The bottom line represents the current CV three-year test cycle that might (and probably is) be too long in many instances.

An automated tool-based test cycle is likely the most reasonable and cost effective way to handle a changing environment when many systems are involved. The configuration audit along with random vulnerability testing is considered cost effective method that can reduce the potential for vulnerabilities as a result of technology improvements. However, the pressing need is to determine in real time if a potential problematic change has occurred that will trigger the need for further investigation. This can be accomplished with the proposed “snapshot” type test approach.

In all cases, the primary consideration for accreditation personnel is to limit the need for testing at a particular user’s site. If it were possible to get the needed information immediately from a remote activity, then that would be the most valuable information for the accreditation team. Failing this, the ability to proactively look for changes using a cost-effective remote approach is a practical solution.

Approach to the Study

The study had several facets including: discover which available remote or local test could identify a system change, evaluate the possibility of a potential vulnerability being detected only by someone at the host, and determine how this potential vulnerability might be detected using a remote means. Additionally, there was the need to determine which test would be most suited to reduce the time that a new change would go undetected after an initial accreditation, and also which tool would allow the most effective use of limited resources. Again it is important to point out that changes are considered trigger events, and that a change does not necessarily mean the presence of a new vulnerability.

Initial research identified two significant detection problems. First, the lack of a remote test that would detect the presence of a major new application, in particular the presence of a previously unidentified web site. Secondly, the need for a tool that would provide change information on NT systems. For example, the fact that a former UNIX server has been replaced by a NT server is a significant change.

To address the web issue, either a test tool that could identify web vulnerabilities is necessary, or a tool that would detect the presence of a web site could be used. Identifying a new web site would allow either the local security representative or the accreditation team a chance to evaluate potential web-based vulnerabilities directly at the host that may not have been evaluated for this threat previously.

Three mitigation techniques will potentially solve the new web site identification problem. Acquiring a remote test tool that can be operated in a web test mode would at least scan for web vulnerabilities remotely. However, based on our limited knowledge, such a tool normally would not find new web sites that do not have any vulnerabilities. The second approach is similar. Write or find a small program to scan an IP range for open ports 8080 or 80. This approach has the flaw in that it won't find sites that have been hidden by using another port. The third approach is to develop a search engine that will scan for all html sites that are within a range of addresses.

For the NT issue, if we are simply looking for an operating system change, there are certain indicators that can be used similarly to the web site indicators. Port 139 (Net Bios Session Service) is not active on UNIX based systems. Additionally, port 135 (Location Service) might also indicate a NT system. Therefore, the same program that scanned an IP range for a web server should also be able to scan for an operating system indicator. If this does not work, a remote ftp attempt would allow the tester to determine the operating system type. Another approach would use the available remote vulnerability test tool.

Suggested Methodology

Once an approach to identify indicators was determined, a search for a port scan freeware tool was undertaken. The intent was to find a tool that could be applied to satisfy only

the needed requirements. While port scanning is an integral component of various vulnerability test tools, only the scanner component itself is required for the proposed test approach. At least one tool with potential to meet CV audit requirements was identified.

Strobe [4]

Strobe stands for Super Optimized TCP Port Surveyor. Strobe is freeware that can be used by subnet site system administrators or by the networking system administrator to verify services on hosts connected to the network. It can also be used as a network/security tool that locates and describes all listening TCP ports on a (remote) host or on many hosts in a bandwidth utilization maximizing and process resource minimizing manner. Strobe approximates a parallel finite state machine internally. In non-linear multi-host mode it attempts to appropriate bandwidth and sockets among the hosts very efficiently. This can result in appreciable gains in speed for multiple distinct hosts/routes.

On a machine with a reasonable number of sockets, Strobe is fast enough to easily perform port scanning of entire sub-domains. Strobe can also survey of very large and complex networks quickly from a fast machine located on the network backbone, provided the machine in question uses dynamic socket allocation or has had its static socket allocation increased very appreciably.

Strobe (1.03) Synopsis and Options

```
strobe [ -vVmdbepPAtnSilfsaM ] [host1 ... [hostn]]
```

The following options are available for use with Strobe V 1.03:

-v Verbose output.

-V Verbose statistical output.

-m Minimize output.
Only print hostname, port tuples.
Implies -d. Useful for automated output parsing.

-d Delete duplicate entries for port descriptions. i.e use only the first definition.

-s Statistical information describing the average of all hosts surveyed is sent to stderr on completion.

-q Quiet mode. Don't print non-fatal errors or the (c) message.

-d Display only the first description in the port services entry file (Cf. -B).

-o file
Direct output (but not any messages which can be affected by -q) to file.

-b number
Beginning (starting) port number.

-e number
Ending port number.

-p number
Port number if you intend to scan a single port.

-P number
Local port to bind outgoing connection requests to. (you will normally need super-user privileges to bind ports smaller than 1024)

-A address
Interface address to send outgoing connection requests from for multi-homed machines.

-t number
Time after which a connection attempt to a completely unresponsive host/port is aborted.

-n number
Use this number of sockets in parallel (defaults to 64).

Strobe attempts to figure out if the number is greater than the quantity of available sockets at any point in time -- and if so, Strobe only uses the amount found. On some UNIX implementations such as Solaris, this appears not to work correctly and the user may have unusual errors such as NO ROUTE TO HOST when the socket ceiling is reached. Strobe may not be the only process running on the system desiring a socket and resource contention may occur. Having Strobe pilfer all the spare sockets away from inetd and other daemons and clients isn't a good idea as it could stop all new incoming and outgoing connections.

-S file
Change the default port services description file to file. Note that if the -S option is not specified, port services are loaded from one of the Strobe specified services,
/usr/local/lib/strobe.services, or
/etc/services.

-i file
Obtain hostnames to Strobe from file rather than from the command line. Note that only the first white-space separated word in each line of file is used, so one can feed in files such as /etc/hosts.

If filename is '-', stdin will be used.

-l Probe hosts linearly (sequentially) rather than in parallel. The actual ports on each host are still checked in a parallel manner (with a parallelism of -n (defaults to 64)).

-f Fast mode, probe only the TCP ports detailed in the port services file (see -S).

-a number

Abort and skip to the next host after ports to the indicated number have been probed and still no connections have occurred. Due to the parallel nature of the probing, reply packets for n+m may return before those relating to n. What this means is that ports > a certain number (n) may be probed. If Strobe identifies a connection on any one of these higher ports before it has negated all possibility of a service listening on ports <= number (n), then despite the fact that all ports up to and including n may turn out to be connectionless, Strobe will `abort the abort'. This is considered optimal, but not unusual behavior.

-M Mail a bug report, or TCP/UDP port description to the current source maintainer.

Strobe Examples

- `strobe -n 120 -a 80 -i /etc/hosts -s -f -V -S services -o out`

Strobe all entries in /etc/hosts (identical ip addresses are skipped automatically) using 120 sockets in parallel, but only check the individual TCP ports mentioned in services. If the user has probed up to port 80 on a host and still not identified a connection, then skip that host. Display speed/time statistics for each host and for the totality of hosts to stderr. Place the regular output in out.

- `ypcat hosts | strobe -p 80 -t 2 -A 203.4.184.1 -P 53`

Strobe all hosts in the host's YP/NIS-table for WWW- servers. Use a timeout of two seconds. Set the source address to the 203.4.184.1 interface. Make all connection

requests appear to come from port 53 (DNS).

Strobe performs no other security functions and does not verify route blocking against UDP or TCP Handshake sequence guessing one-way IP spoofing attacks.

A Proposed Configuration Audit Test

Various proactive vulnerability test tools use a port scanner to initially identify which ports to attack. While Strobe test results effectively perform this function, the proposed configuration audit test requires a baseline. Therefore, the real test for changes is not the Strobe output itself, but the comparison of sequential Strobe test results. These results provide both the initial baseline snapshot, and the ongoing history of configuration change snapshots for all systems tested. Writing a script that will append the Strobe to a file on a monthly basis allows the system administrator to keep track of all services currently running. Each month before overwriting the old output with the new output, a comparison can be made to check for the size of previous month's file, to the current file. If a difference occurs, the file is not overwritten, and an email message alerting the administrator or tester can be sent.

The tester would then compare the two outputs, either manually, or preferably through the script enhancement, to determine if new services have been enabled, or if existing services have been modified or augmented. Although formal testing of the approach has not been performed, it is expected that most of the following Table 2 change indicators would be identified using this technique. Examples of expected results are shown in Table 3.

Change Identifiers¹

- 1. Operating System Change**
--OS Upgrade shows up as a new open ports
--Could also automate SMTP server lookup for version change
- 2. Primary Application**
--shows up as a new open port
- 3. Adding Another Major Application**
--shows up as a new open port
- 4. Adding Another Service**
--shows up as a new open port
- 5. Logical/Physical Change¹**
--would only detect a new set or ports
- 6. Equipment Upgrade**
--mostly undetectable¹
- 7. Security Application Change**
--likely shows up as a new open port

Table 2 – Change Identifiers

a file that did not exist. The baseline snapshot only technique keeps the network's database of current configurations simplified.

Testing/Administrative Approaches

External security testing should be performed on a recurring basis. Sites with significant growth need to be regularly tested for potential configuration problems. Testing serves several purposes:

- External testing with a notification process.
- Identify systems and services running at networked sites.
- Probe random testing without notification to determine if sites monitor their logs.
- New Web server identification testing.
- Identify penetrated systems that have been re-configured by an attacker.

A changed or added IP on a network would be immediately identified during testing since there would be an attempt to overwrite

<i>Change Identifier</i>	<i>Port Example</i>
Operating System Change	<ul style="list-style-type: none"> • NT will have ports 139 or 135 for OS Upgrade • UNIX's root privileged ports are 0 – 1023
Primary Application	DNS service = dns port 53/udp NetBios Name Server = netbios-ns port 137/tcp
Adding Another Major Application	Oracle = oracle port 2005/udp
Creating Web Server	Open port 80 or 8080
Adding Another Service	Time Server = timed port 525/tcp
Security Application Change	AFS/Kerberos Authentication Service = afs3-kaserver port 7004/tcp

Table 3 – Change Identifier Port Examples

External Testing Cycle

A diligent, attentive security person (or staff) should be used to perform regular external testing. Expected growth in network assets and classified information access should drive the need for an automated preventive configuration control mechanism to be implemented. Once implemented, there are a number of steps the security person needs to take in order to perform Strobe configuration audit testing.

- A tester needs access to the DNS server and a complete POC (point-of-contact) list that contains: TCP/IP address, machine type (Sun, NT, etc), hostname, POC name, email address, and phone number for each registered system.
- From the DNS information, create a master file of all TCP/IP addresses.

Example: 199.199.142.1
hostname.sys.mil
trigger.sys.mil
dolphin.sys.mil

- Strip out hostnames in master file. Create blocks (smaller files) based on total number in master file. Example: master file equals 10,000 hosts. To test all systems in a (3) month period, approximately (167) systems would have to be tested daily or (84) systems daily over a six-month period based on an average of (20) workdays a month.

Probe Configuration Audit Testing

The purpose of probe (configuration information gathering) testing without notification is to:

- Check for configuration changes that have not been reported.

- Determine services running (telnet, ftp, http, routed, etc.) on boxes at random sites.
- Determine what sites are reading their system logs, and reporting unscheduled probes?
- Obtain an estimate [5] for the number of web servers running at sites.

Identifying Services

After accreditation and testing, a system is allowed access on a network. Using Strobe, a security tester can take a configuration snapshot to identify TCP ports that are open for services. Over time new applications (web servers, anon ftp etc.) may be added to a server on the network. Strobe helps the tester identify where changes in services have taken place. Running Strobe on a regular basis also helps individual sites (system administrators) track these changes. Strobe does not gain access to a system but its results could be used as part of the process to identify where potential vulnerabilities might be located. Being aware of what (TCP ports) services are open to outside sites is important information for configuration auditing as well as for identifying port vulnerabilities.

Random Unscheduled Probes

If unauthorized personal are probing protected networks, system administrators should be reading their system logs and reporting the findings immediately. Random, unscheduled (no prior notification) probes will determine who and what percentage of site administrators are reading system logs and reporting these probes. If administrators don't read logs (system and web logs), there is the potential that system resources can be compromised for extended periods of time. The potential even exists that information could be distributed (by accident) to those who have the proper clearance but "not a need to know".

Sample Test

Recording the number of web servers running on a network helps system administrators to identify potential vulnerability locations. Classified information will continue to be distributed (possibly significant growth) in html format. For simplicity, web servers have become a preferred method to post (exchange) information. Maintaining a uniform configuration of web server access is difficult.

Unscheduled Probe Testing Cycle

This section describes a proposed configuration audit test approach by actively checking how diligently system administrators are at checking their logs for probes.

- a. An experienced security person (ST&E person) should be assigned for random probe testing.
- b. The ST&E person needs access to the DNS server.
- c. Using DNS, create a master file. Strip out hostnames. Then create smaller files based on the total number in master. Example: if master equals 1000 subnets (199.199.24.) or 1000 networks (sites), choose one out of every ten.
- d. Strobe is a very fast scan tool that runs on many different platforms. Strobe is run from the command line (No GUI).
- e. Results should be kept in a database. These results should determine, who, how many, which sites report or otherwise noted the probes. Phone calls and email should be placed to site's POC (system administrator) after a set time (one or two weeks) to discuss:

- Did they notice the probes? (Y/N)

- Did the site report the probes? (Y/N)

Inform POC/system administrator how to report probes and to whom?

- Do sites keep system logs for more than a month? (Y/N)
- Do sites check and/or run logging on web servers? (Y/N)

Further Notes on a Random Test Cycle

After probing 10% of the networks, the tester should go back to one specific server (on each network) running multiple services. These services include: telnet, http, routed, ftp etc. The tester should attempt access with general hacker approaches, telnet <system name, root, guest > anonymous ftp, cgi-bin attacks on port 80, etc. Contact the POC (admin) 5-7 days later (phone or email) to find out if the probing attempts were logged, reported, or even noticed. If the attempt remained unnoticed, this would be an indication that further more specific vulnerability testing is warranted.

Conclusions

There is a pressing need by organizations to quickly perform remote real-time security configuration audits is such a way that security related change information is identified. The proposed methodology using a variation of a readily available test tool, Strobe, will solve this need in a cost-effective and easy to use manner. Additionally, the results of longer term regular testing will provide the database necessary to validate time periods currently specified for the compliance validation of systems under Phase 4 of the DITSCAP.

References

- [1] Department of Defense (DoD) Information Technology Security Certification and Accreditation Process (DITSCAP), DoD Instruction 5200.40, December 30, 1997.

[2] System Security Authorization Agreement (SSAA) – Accreditation document required within the DITSCAP process (see enclosure 6 of DITSCAP).

[3] DRAFT Compliance Validation Implementation Plan, SAIC for DISA D25, October 10, 1998.

[4]
<ftp://coast.cs.purdue.edu/pub/tools/unix/strobe/>
or from
<ftp://suburbia.net/pub/strobe.tgz>

[5] Due to the possibility that a hidden Web service was installed but not necessarily used on the initially accredited system, Strobe would still detect an open port but not that it was used for Web services.